Programming Assignment 3: SDN and Ryu

Roshan Raj : 2019CS5037

November 1, 2023

1 Part 1: Controller Hub and Learning Switch

1.1 Code Implementation

In this part, we implemented two key components: the Controller Hub and the Learning Switch using Ryu, a software-defined networking (SDN) controller. The Controller Hub behaves as a basic switch, redirecting all incoming traffic to itself and forwarding it to all switch ports except the incoming one. The Learning Switch, on the other hand, operates like a typical Ethernet switch but dynamically installs flow rules based on MAC-to-Port mappings it learns from incoming traffic.

The main components of our code include:

- `controller_hub.py`: This file contains the implementation of the Controller Hub.
- `learning_switch.py`: This file contains the implementation of the Learning Switch.
- Flow rule management using Ryu's OpenFlow-like APIs.

1.2 Design Decisions

1.2.1 Controller Hub

The Controller Hub implements basic switching behavior by forwarding traffic to all ports except the incoming one. We use Ryu's OpenFlow APIs to create a table-miss flow entry to send all traffic to the controller.

1.2.2 Learning Switch

The Learning Switch learns MAC-to-Port mappings from incoming traffic and dynamically updates flow rules. It uses flow entries to forward traffic based on these learned MAC addresses. We handle ARP requests and respond appropriately.

1.3 Test and Observations

We conducted ping tests for both Controller Hub and Learning Switch scenarios and reported latency values. The Learning Switch should show lower latency as it uses dynamic flow rules. We ran a throughput test between hosts h1 and h5, and explained the differences between the two scenarios. We executed `pingall` in both cases and reported the installed rules on the switches.

2 Part 2: Firewall and Monitor

2.1 Code Implementation

In this part, we implemented a Firewall and Monitor within the network. The primary objective was to restrict communication between specific hosts (e.g., H2 and H3 with H5, H1 with H4) for security reasons and count all packets coming from host H3 on switch S1.

The key components of our code include:

- `firewall_monitor.py`: This file contains both the Firewall and Monitoring functions.
- Management of firewall rules to block specific host communication.
- Counting packets from H3 on S1.

2.2 Design Decisions

2.2.1 Firewall

We added firewall rules to block communication between specific hosts based on predefined rules. The firewall functionality leverages MAC address filtering to control traffic.

2.2.2 Monitoring

We counted packets coming from H3 on S1, providing visibility into network traffic. The Monitoring functionality uses the Learning Switch's capabilities to track packets.

2.3 Test and Observations

We ran `pingall` to observe the results of firewall rules in action and reported the installed rules on switches. We considered how to minimize the number of firewall rules on the switch. One way to minimize rules is to create wildcard rules that match a range of addresses. We discussed how to implement firewall policies in real-time without interfering with pre-existing rules. A solution would be to dynamically manage rules and have a mechanism to add, modify, or remove rules as needed.

3 Part 3: Load Balancer

3.1 Code Implementation

In Part 3, we implemented a round-robin Load Balancer on switch S1 to evenly distribute requests from hosts (H1, H2, and H3) to servers (H4 and H5) using a virtual IP address (10.0.0.42).

Key components of the code include:

- `load_balancer.py`: This file contains the Load Balancer implementation.
- Handling ARP requests to respond to virtual IP requests.
- Implementing round-robin load balancing between servers.

3.2 Design Decisions

3.2.1 ARP Handling

We respond to ARP requests for the virtual IP address (10.0.0.42) with the load balancer's MAC address. We also cache the MAC addresses of the servers to facilitate load balancing.

3.2.2 Load Balancing

We distribute traffic in a round-robin fashion between the servers (H4 and H5) using flow rules. The load balancer alternates between servers for each request.

3.3 Test and Observations

We had hosts H1, H2, and H3 ping the virtual IP (10.0.0.42) and observed the installed rules on switches. We discussed additional steps to implement load balancing policies considering the load on servers. Options include tracking server load, implementing weighted round-robin, or using a more complex load balancing algorithm.

4 Conclusion

In this assignment, we successfully implemented a Controller Hub, Learning Switch, Firewall, Monitor, and a Load Balancer using Ryu, an SDN controller. These implementations achieved the specified objectives and allowed for the restriction of communication, monitoring network traffic, and load balancing of traffic in a controlled manner. We also discussed ways to optimize and enhance these functionalities for real-world use cases.